Annual Dues..................$12.00

6/84

L.I.S.T. Group

LONG ISLAND
SINCLAIR TIMEX
GROUP

Centerport, N.Y. 11721

L.I.S.T.ING

I. MEETING NOTES -JUNE 3, 1984

A longish 4 hour meeting at Idle Day Drive involved:

a) Paul D. tried to log on to compuserve but, while we could get in, the password (which doen't show on the screen when typed) must have been mistyped and the system threw us out. Rather than hold up the meeting (or run up the phone bill) we gave up after 3-4 tries. Rest assured, the MD-2 Modem and ZCOMM do work.

b) Nazir P. showed us his translation of the ZX Spectrum arcade game "Galatoids" (Public Domain) and the numerous changes in ROM calls, particularly the use of the "blank" portion of the Spectrum ROM, which were necessary to adapt it to the 2068.

c) Paul D. demoed "Transylvanian Tower", a Spectrum adventure-maze game sent to us for review by Richard Shepherd software. A considerable amount of work (detailed inside) was necessary to get even this BASIC program to run.

d) Rich B. demoed Vu-file from Psion and its included "Gazeteer" program. Speed seems comparable to Tom Woods Pro/file, but formatting may be a little more rigid.

e) FIRSTLOADR - ZX81 upload program from SYNTAX, was tried several times, without success. Heinz H. feels his "smoother", to be demoed next meeting, may be the answer. Volume setting is clearly critical.

f) Group purchasing was dicussed, but little consensus or advantage was found.

g) A discussion of "piracy" (read theft) lead to a unanimous voice vote that such was against group policy, the possible exception being companies which have folded. More discussion next time.

II. NEXT MEETING IS SCHEDULED FOR 7 JULY (SATURDAY) AT 3:00PM

Location: 9 Dartmoor Drive, East Northport - Call 368-9088 for directions, ask for Heinz. Please bring any special software, hardware items you'd like to demo. Remember too, that many of us are tyros. That simple game program you wrote last year and which is now too slow or BASIC for you, may be ideal learning aid for the beginner (especially if you've documented it). Also, we have begun to create a "library" tape of public domain (or member donated) software and of course, would like to share such programs between members.

INDEX TO ISSUE-JUNE 84

Your reviews, programs, comments, hardware projects, etc., are eagerly solicited for publication in LISTing.

*OR REPRODUCED,

Sample copy sent upon receipt of large SASE.

Copies provided on exchange basis with other bona fide user groups.

L.I.S.T.ing is published monthly by LIST (Long Island Sinclair Timex) Group
a not for profit users group

FOR BEGINNERS ONLY: FIRST LOADR

An outstanding achievement by David Ornstein and a great public service by SYNTAX, FIRSTLOADR lets you upload your TS1000 BASIC programs to the 2068. Unfortunately, several of the beginners (and more advanced programmers too) have had trouble understanding, entering and/or using FIRSTLOADR. Based on the questions most frequently asked, we can offer the following tips:

1. The data shown in Syntax (December 83) on page 14 is a translation table. The machine code (MC) program,(P15) which runs from 45000 to 45234, uses this table to convert ZX/TS code to its 2068 equivalent. That is, each number in the table is the CODE of one of the 255 characters you can use. Several characters are not directly translatable, and these have been replaced by "32" or a blank space. Check the character tables in the backs of both manuals for a comparison. Right now, CLEAR 43999 & NEW.

2. To get these numbers into the computer, look at Fig.1 in this months article on Transylvanian Tower. Lines 100 to 140 serve as a rudimentary INPUT routine which lets you put in 10 bytes at a time. If you GOTO 100, you'd be expected to put in a value for V. Enter 44000 for example, to begin entering data in the look-up table. The program will print the current memory location and wait for your input, 't'. Again if you were just starting out, you would enter 32. Next you'd enter 130,129 etc. ending with 143 (the 10th byte).

3. Now to check what you ve done so far, break into the program (or add a line-145 STOP) and GOTO 205. Again you'll be asked for a "Start". Enter 44000 and you'll see the bytes you just entered arranged just as they are in Syntax. The program will list 300 bytes (which can be changed by editing line 210), so at this each stage you may want to BREAK after you've seen what you want.

4. Look at that listing. If any numbers were entered wrong, POKE the correct value directly. That is, if you see:

   ```
   44000   32   130   129   133   136
   ```

   You know the 4th value is wrong (it should be 131). So now, in immediate mode, POKE 44003, 131.

5. Repeat the process of steps 2 through 4 until you reach 44255. Next, enter the actual machine language program by going to 100 and letting v = 45000. Fill any excess bytes with 0 or 201.

6. Triple check the whole entry table and program by GOing to 205 (by the way, lines 10 to 40 and 300 are not needed in this application) and camparing what you have on the screen with the listing in Syntax. Correct any errors with direct POKES.

7. SAVE "FIRSTLOADR" CODE a couple of times, for insurance. (You might even want to do this before step 6, just in case you mistype a POKE and crash the machine.)

8. Now, cue up a ZX81 tape (use a 2K or less program for testing purposes) as described in Syntax.. Volume is critical and about all I can tell you is that the program data should look like red and black lines in the border somewhat similar to the yellow and blue of a 2068 program in width. Too low or two high a volume will give nothing or just occasional flashes. Make sure there is no BASIC program is the machine when you RAND USR 45000.

9. Enter RANDOMIZE USR 45000, play the tape, and wait for the 0 OK. Note that you can stop the process, anytime, by pressing the 'BREAK' key. Note particularly that Ornstein means just that key - not Capshift and BREAK.

10. Go through your program as described in Syntax, note changes required and SAVE it before you make them.

My first successful firstload was "Memory" a simple 2K Simon Says game from the Mixed Game Bag II tape. The only changes were to redefine variable X, the place to start in the character table, from 28 to 48 (line 140), change the "SAVE" line to add LINE for autorun and add OR a$= "y" to cover the lower case option for replaying the game.

MOONBUGGY:

Moonlander - line 630 the > sign should be <> ?

A CALL ABOUT A PARROT

John Cox called from the West Coast to find out more about RIST's Parrot. He couldn't get through to RIST, but fortunately, they had sent me a schematic. I told John what the missing chips and software on his second-hand kit version were and hope he does well in assembling it: IS RIST still in business?

LI.S.T. Group

@ Copyright 1984
P.J. Donnelly

BREAKING INTO THE TRANSYLVANIAN TOWER

Transylvanian Tower, by Richard Shepherd Software, is a 3-D MAZE adventure game written entirely in BASIC. The game begins with an external view of the "tower" and 3 brief introductary screens of instructions. Do read the instruction booklet which comes with the cassette though, before beginning play. You'll get a chance to do this in the 2 minute delay, while the computer sets up the first (and each succeeding) maze.

The game is played on 5 levels of the tower, each level is an 8 X 8 maze of interconnected rooms, some of which hold treasures (e.g. a silver cross) and many of which hold deadly vampire bats. Your task is to proceed from the bottom or 1st level, where there are no treasures or bats, up through tougher and tougher challenges to level 5, kill Count Kreapie and obtain his treasure.

In the first level, and in each succeeding one, you are presented with a line drawing showing 3 walls, ceiling and floor of the room in perspective. Doors, windows and objects are also represented. If there is a bat in the room, he will fly around the ceiling until you either kill him with your gun, or flee,or are killed by him.

The game is a very good example of what can be done with SINCLAIR BASIC and the Spectrum/TS 2068. The graphics are very good for BASIC and the pace is adequate, if slow during certain phases. Those slow phases are the two minute wait between levels and the agonizingly slow drawing of your situation map (which takes about 30 seconds).

The author has protected his software by jamming the machine stack, so don't try to break the program while it is running or you'll restart your machine. Note too, that the Spectrum version will not run on a TS 2068, without significant modifications (see companion article). We found the difficulty of the game on a whole adequate, although, firing your gun was not in the least bit convenient.

Overall I'd give TT a rating of 7 out of 10. It is an impressive achievement for BASIC, but could have been better documented (e.g. the instructions don't tell you how to fire your gun-use the ZERO key). It's price - £6.95 (about $11.00) is about right for the U.S. market and a reasonable value for the money.

Suppliers	Item	Cost
Best's	2068	$98.32
47th St Photo	"	$99.95
Timex	"	$99.95 (No Tax)
Timex	TS1000/16K 50 SO FT	$49.95
Zebra†/Sunset	Modem	$119.95
Fordham Radio	Joysticks (BP)	2/$8.00
Radio Shack	26-1332 * Thermal Paper	2/$3.95

† Bob M. has one and is hooked up to the Source
* Thanks to Rich B for this tip.

THE TOWER

Spectrum BASIC is much closer to ZX81 (TS1000), BASIC than that of the 2068 and it was this similarily, which enabled me to rework Transylvanian Tower (TT) for the 2068. TT is SAVED as CODE starting at 23552 for 36,500 bytes. The Spectrum uses the area from 23552 through 23755 to store the system variables and the BASIC program can begin anywhere after that. In this case, fortunately, BASIC begins at 23755.

Your TS 2068 on the other hand, while keeping system variables at 23552 through 23755 (though they differ in function after 23733), also has a number of other housekeeping chores tucked away in the 16-24K area. Most notable of these is the machine stack which normally sits (with one display file - mode 0) at 24576 to 25088. Other items, Chans, etc. are also kept in this area, so having BASIC there is not such a great idea.

In order to find out how a BASIC program is configured, we need to look at that systems variables file. That is easy enough to do if we use the LOAD ""CODE STARTLOC, BYTES command. Here STARTLOC is the location we want the TS 2068 to start loading code into and BYTES is the number of bytes to load. If we don't know how long a program is we can leave BYTES blank.

For the purposes of examining the systems variables (203 bytes) and perhaps a little of the BASIC program then, we do the following:

1. CLEAR 28551 & NEW - This sets RAMTOP to 28552 and leaves whatever we put above it isolated from the BASIC System.
2. LOAD"" CODE 28552 - which will load the tape into 28552 and beyond. To save time while experimenting, you might want to stop the tape after 30 seconds or so. The whole program won't LOAD, but you should get the first 1K bytes, more or less.
3. LOAD or type in at least lines 205 through 295 of the program shown in Fig.1. Then GOTO 205. You'll be asked for the starting address. Respond with 0 or 28552. The program will then list the code you just loaded, but will offset the effective addresses to reflect the locations the bytes originally had. Typical output is shown in Fig.2 (can be obtained either by COPYing the screen or OPEN #2, "P", GOTO205, CLOSE #2).
4. Prepare a chart, like Fig.3, which lists the systems variables and calculate the values of the codes obtained in step 3. The really important addresses are ERR SP, LIST SP, virtually all the bytes from NEWPPC (23618) to MEM (23657), RAMTOP 237301-1, and the CHANS data; 23618 to 23755. Remember that the values are stored LSB first. As an example, to find out where the program starts, let's investigate and use the variable PROG. (see PP 262ff of TS 2068 manual)
5. Calculate the value of PROG. From Fig.3 we see this is:

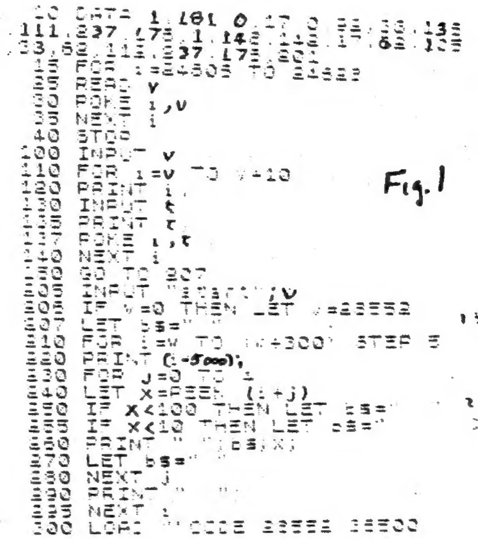
23635	contains	LSB = 203	203
23636	contains	MSB = 92 X 256=	23552
PROGRAM STARTS AT			23755

   Does it? Take a look at 28754. (offset 5000 from 23754) We see the 128 (80H) we are supposed to see before the program starts (see Page 255 of 2068 manual). Next comes:

   0 1 77 0 244 50 etc.

   Line numbers are stored in the normal way in 2 bytes, so the program starts at line #01. The next two bytes say the rest of the line is 77 bytes long. Look at 28836 (77 + the 4 bytes we've just looked at). We see code for line 02. You could interpret each byte of code to determine what the program does, but there is an easier way.
6. The PROG we've been looking at is really at 28635 and the actual value, under which the system is operating is at 23635. We know that the program begins at an 28755 so we can POKE that data into PROG and fool the system for a while. 28755 = 83,112 So:

   POKE 23635,83
   POKE 23635,112
7. LIST your program. The very beginning of the first line is garbage like Fig.4, but you can figure out from Fig.2 or 3 what those few bytes were (in this case it was 01 POKE. Try COPY or LLIST if you want a more permanent record. You could now simply type in the program and, given that the author didn't enter any variables as direct commands, it would work. There is an easier way, however.



```
 10 DATA 1,181,0,17,0,[illegible]
111,237,175,1,148,[illegible],62,138
 33,88,[illegible],237,176,201
 15 FOR i=24305 TO 24333
 20 READ v
 30 POKE i,v
 35 NEXT i
 40 STOP
100 INPUT v
110 FOR i=v TO v+10
120 PRINT i
130 INPUT t
135 PRINT t
137 POKE i,t
140 NEXT i
150 GO TO 207
205 INPUT "start?";v
206 IF v=0 THEN LET v=28552
207 LET b$=""
210 FOR i=v TO (v+300) STEP 5
220 PRINT (i-5000);
230 FOR j=0 TO 4
240 LET x=PEEK (i+j)
250 IF x<100 THEN LET b$=" "
255 IF x<10 THEN LET b$="  "
260 PRINT " ";b$;x;
270 LET b$=""
280 NEXT j
290 PRINT " "
295 NEXT i
300 LOAD ""CODE 28552,36500
```

Fig. 1

15

2b>

1b>

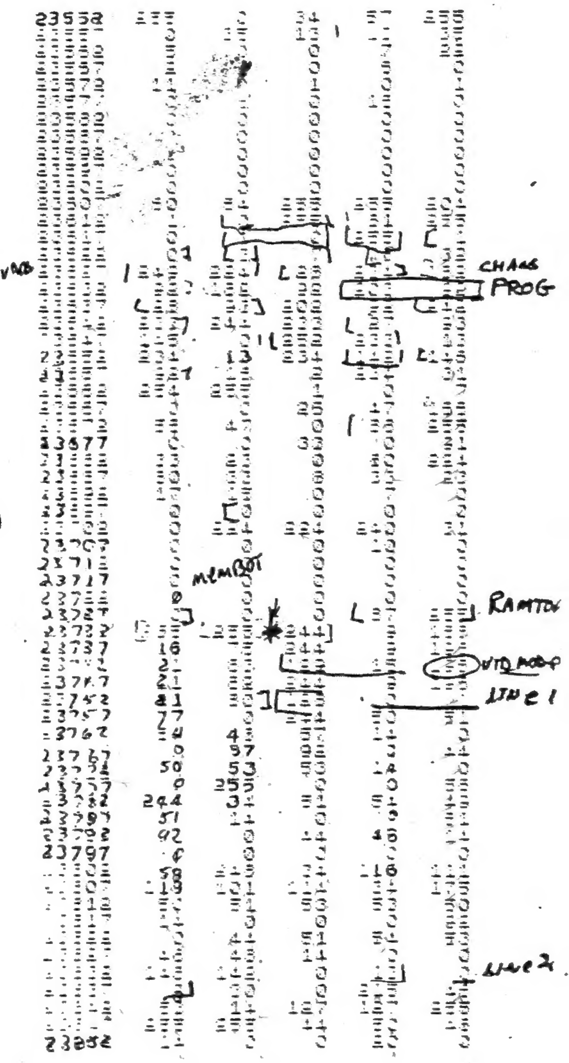


Fig 2

The POKE 23613,0 lines, which mess up the error handling (e.g. causing a system reset when you try to "break" the progr for example), can be replaced by POKE 23620,0 which is innocu enough, and maintains program length) or simply eliminated. If they are eliminated, the program becomes shorter and faster.

8. Turn off your machine and perform steps 1 through 4 again. Remember that we have loaded the code 5000 bytes higher than when it was saved. If the program doesn't dynamically define too many new variables we can stick with this 5000 byte offset. TT does however, and if we redefine all the systems variables to points 5000 higher we end up running out of memory later. We'll move the program down another 1792 bytes to allow it to grow.

   Now look at the old system variables VARS (28627) to MEM (28656).* In each case, we must redefine the location to which they point. VARS, for example, points to 62593 (which means it was origninally at 57,593). We want it to be at 60801 (1792 less) so we change the <u>old</u> values:

```
POKE 28627, 129                  129
POKE 28628, 237   237 X 256 =60673
                             60801
```

   The same must be done for most of the rest of these variables. A particular exception, however is CHANS (& CURCHL) which, as you can see, point to 23734. Looking at Page 255 in the manual, we want the CHANS data (those bytes from 23734 through 23754 in the original program) to be just below the PROGRAM area. There are some 21 bytes and they will start, therefore, at 26942, while PROG will end up at 26963.

   A little confusing at this point, so lets recap:

   a) We loaded the code 5000 bytes higher through originally intended.

   b) That code contains the system variables and the program.

   c) Still fooling around above RAMTOP (28552) we poked new values into the systems variables which would reflect the relative offset of 5000 (actual) less 1792 (desired) or 3208. This was to give more space for VARS to grow.

   d) If we could now move this corrected code where it belongs: i.e., the variables from 28552 through 28733 down to 23552 and, simultaneously, the program (& CHANS) down to 26963 (26942 for CHANS), we would have a runnable program.

9. The first few lines of Fig.1 do that for us. They are a machine code LDIR routine. All that needs to be done now is to RUN the little program and then:

```
LET A = USR 24606: STOP
```

   The information will be put where it's supposed to be and the program can be run by typing

```
CLEAR
GOTO 2.
```

   The machine Code is given in Fig.5

You've now got a working copy of Transylvain Tower. The same technique should work for most such Spectrum BASIC programs.** Be careful though, to check each variable, one at a time, as they really can point anywhere. TT was "easy" because everything was in order. Other programs may not be that way. Also note that you still have to do a little BASIC "cleaning" of the program. Lines 1 and 7720 contain the statement SAVE "tower" CODE 23552, 36,500 - which must be changed to reflect your new configuration. Also, the keyword "STICK" will appear wherever the graphic "I" exists on the SPECTRUM.

If you've developed on this system, let us know and we'll add your comments to this report. As an example, lines 100 to 140 allow input of values to be POKED, starting at address "U". A more sophisticated program might ask for a starting address (e.g. 28635) and ending address (e.g., 23653) and an offset and then automatically adjust the bytes at those locations for you.

<u>THE MACHINE CODE</u>

LDBC with 181 - # Bytes to be Transfered	01,181,0
LDDE with Destination address (23552)	17,0,92
LDHL with Source of Code (28552)	33,136,11
LDIR Transfer the code	237,176,
LDBC with 36,500 bytes	01,148,142
LDDE with Destination 26942 (CHANS & PROG)	17,62,105
LDHL with Source 28734 (original CHAMS &PROG)	33,62,112
LDIR Transfer the data	237,176
RET Return to BASIC	201

Fig. 5

LONG ISLAND
SINCLAIR TIMEX
GROUP

FIG. 3

LOCATION	NAME	CONTENTS (LSB,MSB)	CONTENTS DECIMAL	DESIRED CONTENTS DECIMAL	DESIRED CONTENTS LSB,MSB
23552	K STATE	N/A	65280		
23613,14	ERRSP	0,255	65280	24832	0;97
23615,16	LISTSP	81,255	65361	24913	81,97
23627,8	VARS	249,224	57593	60701	29,237
23629,30	DEST	28,234	59932	63140	164,246
23631,2	CHANS	182,92	23734	26942	62,105
23633,4	CURCHL*	182,92	23734	26910	30,105
23635,6	PROG	203,92	23755	26963	83,105
23637,8	NXTLIN	28,93	23836	27044	164,105
23639,40	DATADD	202,92	23754	26962	82,105
23641,2	ELINE	242,233	59890	63098	122,246
23643,4	KCUR	244,233	59892	63100	124,246
23645,6	CHADD	27,93	23835	27043	163,105
23647,8	XPTR	N/A	-	-	-
23649,50	WORKSD	252,233	59900	63100	132,246
23651,2	STKBOT	13,234	59917	63125	149,246
23653,4	STKEND	13,234	59917	63125	149,246
23655	BRER	142	N/A	-	142
23656,7	MEM	146,92	N/A	-	146,92
23730	RAMTOP				

* ARS BUF?

1) All of these variables need not be reasigned, it's just safer to do it this way.

O I POKE

```
3350    5: POKE 23613,0: SAV
E "tower" CODE 23552,36500
  2 BEEP .3,20: GO TO 990
 10 PAUSE 0: IF INKEY$="s" THEN
 GO TO 20
 11 GO TO 1231
 20 INPUT V$
 22 IF V$="                 " THEN GO SUB
9999
 23 GO TO 1231
400 POKE 23613,0: BORDER 1: INK
 7: PAPER 1: CLS : INK 7
402 PRINT "The following object
s will be   found around this fl
oor.They     each have a battle r
ating."
 403 PRINT : PRINT "0  Magic Rin
g - enables the          bearer to
 pass through two        solid wal
ls.Vanishes when         used up."
 405 PRINT : PRINT "1  Magic clo
ak - using this the      bats cann
ot see you and so        cannot at
tack."
 407 PRINT : PRINT "2  Magic Ass
le - This allows you     access to
 the next floor          without h
aving killed the         correct n
umber of bats."
 409 PRINT AT 21,0;"Continued"
 410 POKE 23624,3: RETURN
 420 POKE 23613,0: BORDER 1: CLS
: PRINT "3  Floor plan - see wh
ere you        are at any time."
 422 PRINT : PRINT "4  Clove of
Garlic "
 424 PRINT : PRINT "5  Knife "
 426 PRINT : PRINT "6  Mirror"
 428 PRINT : PRINT "7  Dagger"
 430 PRINT : PRINT "8  Sword"
 432 PRINT : PRINT "9  Silver Cr
oss"
 433 PRINT : PRINT "Press ""P""
to pick up an object  in a room
(maximum 3 objects)"
 434 PRINT : PRINT "Press ""I""
to get an inventory"
 440 POKE 23624,3: RETURN
 540 POKE 23613,0: PRINT "You wi
ll now begin to meet the  deadly
 vampire bats.They can be killed
 by using your gun or by  using
a weapon which you may   have a
cquired"
 542 PRINT "You can also move on
 to another room,but always act
quickly to  avoid being destroye
d"
 543 PRINT "You must kill at lea
st 20 bats  in order to gain acc
ess to the  next floor."
 544 PRINT "Also, the bats all c
arry floor  plans to guide them
about the   tower,so if you kill
 a bat then you get a floor plan
"
 545 RETURN
```

Fig. 4

← e

F 4

** It works, for example, on "Everest Ascent".

* OFFSET 5000 bytes

L.I.S.T. Group

@ 1984 P. DONNELLY

TIMEX APPLICATION NOTE: INTERFACING WITH BROTHER'S EP44 TYPEWRITER

EQUIPMENT REQUIRED:
1) TS 1000/ZX 81
2) Byte Back MD-2 Modem with RS232 Port
3) Brother EP-44 portable typewriter
4) 14 pin dip header, wire & 25 PIN "D" connector

Brother's EP-44 hi-resolution, dot-matrix printer uses regular and/or thermal paper (up to 8 1/2" wide), can be battery operated and is a full function typewriter. It also has 4K memory for text and an RS-232 port. Priced between $269 and 300, it is a good buy simply as a portable typewriter. With Byte back's MD-2 Modem/RS 232 port, the EP-44 can serve as your printer.

To make this connection, begin by removing the right hand cover plate of your MD-2 unit. Inside, you'll see the backward mounted 14 pin DIP socket shown in Fig. 3. The numbers on the pins are tricky because of this back-side mounting, so be particularly observant.

Assemble your interface cable using the pinouts shown in Figs. 1,2, & 3. Only 3 wires (and possibly just 2) are needed for this interface, they are:

1) Ground-connect pin 7 of the DB 25 connector to pin 7 on your Byte Back board (use a DIP HEADER).

2) DATA TO THE PRINTER This means the Timex will be transmitting (pin 2) and the EP-44 will be receiving (pin 3 on DB-25)

3) EXTERNAL READY - CLEAR TO SEND - (In fact, we could probably dispense with this one or use RTS). External Ready (pin 20) on the printer tells the Timex's Clear to Send (pin 5 or DIP) that it is ready to listen to data.

Now plug in the EP-44 and MD-2 using your cable. Put the MD-2 on the back of a TS 1000, power up and LOAD or Key in the Byte Back software. Switch the EP-44 on and put it in terminal mode (OFF Line). Enter the configuration shown in Fig as the printer modes. The BASIC software listing supplied, free, by Byte Back was used to generate the character set and message shown below:

```
THIS IS TO INFORM YOU THAT


 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
pqrstuvwxyz{|}~
```

As supplied, you can only use LPRINT, not LLIST & COPY. This also limits you to 32 characters per line for standard text. Byte Back has a program which will give you full features at $29.95.

This system is easy to hook up and quite reliable. I don't know which RS-232 interface chips the Brother uses, but it appears quite rugged and flexible, as is the MD-2 modem. We only tested the EP-44 as a printer, but by connecting the remaining RXD and TXD lines and RTS, CTS, you could use the EP-44 as a "big keyboard".

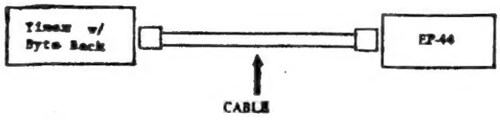
Finally, though we've not 100% sure of this, it may even be possible to tap into the EP-44's 4K of CMOS, battery backed-up RAM. This means you could add 4K RAM and, have "almost" instant load software, as long as it was under 4K.

5-2-84

ByteBack
SINCLAIR TIMEX

1. CONNECTION

EP-44 to be used as an output printer for personal computers.



2. CONNECTING CONDITIONS

SINCLAIR TIMEX

BAUD RATE	300
(BY	)
BIT LENGTH	8
STOP BIT	2

EP-44

BAUD RATE	300
BIT LENGTH	7
PARITY	Z
NEW LINE	CR + LF
CODE	8 BIT
ER	Y

ByteBack
RS-232 Output Plug

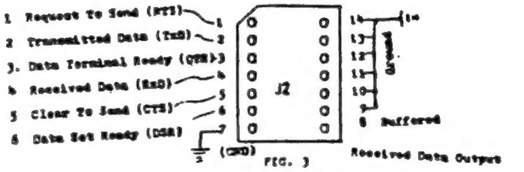
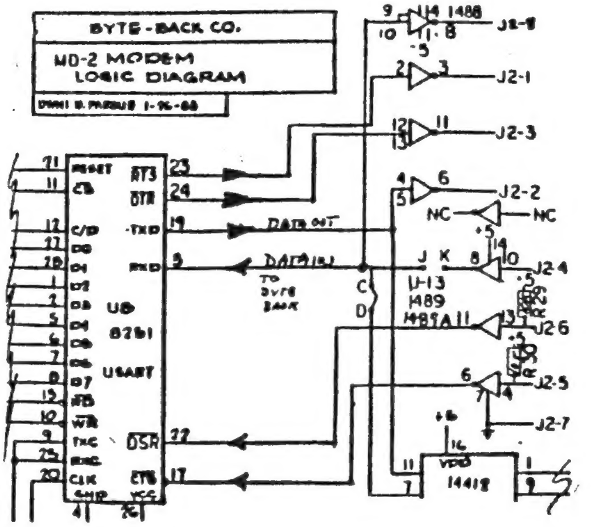


FIG. 3

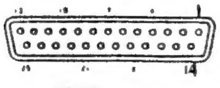
AS VIEWED FROM THE COMPONENT SIDE OF THE BOARD

Note: The socket is on the back or circuit side of the board, but the pins are numbered as shown viewed from the front side.



Connector and pin arrangement

Terminal number	Signal name	Code	EP-connected device	Function summary	Note
2	Send Data	SD	→	Data line sent from the printer to connected terminal	
3	Receive Data	RD	←	Data line sent from connected terminal to the printer	
4	Request to Send	RS	→	Controls transmission carrier ON: carrier output OFF: carrier stop	Normally ON in the terminal mode
5	Clear to Send	CS	←	Controls data transmission ON: data transmission possible OFF: data transmission not possible	
6	Data set Ready	DR	←	Indicates condition of connected device ON transmission/reception possible at connected device OFF transmission/reception not possible at connected device	ON when cable is not connected
7	Signal Ground	SG	—	Provides basic ground potential	
8	Carrier Detect	CD	←	Detects carrier ON receiving access signal OFF not receiving access signal	ON when cable is not connected
20	External Ready	ER	→	Indicates condition of the printer ON printer preparation completed OFF printer preparation not completed	Enables on/off setting selection ON unless otherwise specified.



(RS 232C) connector

NOTE: Hand shaking

(1) Reception

- Reception is possible when DR and CD are ON; reception data is ignored at other times.
- If ER is set, ER line is ON when data can be received, but becomes OFF when busy (print buffer remainder is low and data cannot be received).

(2) Transmission

- Data is transmitted when DR and CS are ON.